

Das bringt uns nicht viel weiter. Die Fehlermeldung und das verstümmelte Wort »COMMODORE« am oberen Bildschirmrand deuten aber auf einen fehlenden DATA-Wert in Block 2 hin. Untersuchen wir also Block 2 einmal genauer. Das Unterprogramm zum Ausdrucken der Werte von I und X am Bildschirm befindet sich ja ab Zeile 1000 noch im Speicher. Wir brauchen daher nur einen entsprechenden GOSUB-Befehl in die zweite Leseschleife einzufügen, am besten gleich nach dem READ-Befehl, also etwa in Zeile 225:

```
225 GOSUB 1000
```

Wir erhalten wieder eine leicht zu überprüfende Liste aller DATA-Werte, diesmal aus Block 2. Bei I=4 fällt uns sofort etwas auf. Am Bildschirm erscheint nämlich

```
I = 4      X = 15.4
```

Das ist die einzige Zahl mit Nachkommastellen, was bei dieser Art der Bildschirmausgabe sofort ins Auge sticht. Wir vergleichen den Wert mit der Eintragung im Originallisting und sehen sofort den Fehler: Wir haben beim Abtippen irrtümlich einen Punkt statt eines Kommas eingegeben. Die Korrektur ist leicht ausgeführt.

Danach löschen wir die jetzt überflüssige Zeile 225 mit dem GOSUB 1000 wieder und überzeugen uns durch einen abschließenden Probelauf vom einwandfreien Funktionieren des Programms.

Natürlich kann man nicht erwarten, daß sich alle Fehler so reibungslos lokalisieren lassen wie in unserem kleinen Beispiel. Gerade bei Fehlern in DATA-Zeilen kann die Suche sich namentlich bei längeren DATA-Blöcken um einiges schwieriger gestalten. Aber bei Programmen mit vielen DATA-Zeilen sind die hier beschriebenen Methoden zum Auffinden von versteckten Fehlern einfach unentbehrlich, wenn man einigermaßen schnell und sicher zum Ziel gelangen will.

Das Optimum herausholen

Zum Schluß soll noch vor einer verbreiteten Unsitte gewarnt werden, die für viele fehlerhafte Programme verantwortlich ist. Gemeint ist das Optimieren von Programmen hinsichtlich Speicherbedarf, Geschwindigkeit, Benutzerfreundlichkeit oder eines anderen Kriteriums.

Vielfach wird dabei versucht, während des Abtippens eines Programmes gewisse Dinge zu ändern, eben zu optimieren. Da werden Variablenamen geändert, mehrere Zeilen zu einer einzigen zusammengefaßt oder andere PRINT-Anweisungen eingebaut.

Da man ein komplexes Programm jedoch nicht so einfach überblicken kann, entstehen durch solche Maßnahmen nur zusätzliche Fehler. Es gibt zwei bekannte Programmierer-Faustregeln zum Optimieren von Software:

1. Optimierte nicht.
2. Optimierte noch nicht.

Die erste Regel kann man auch so ausdrücken: Wenn das Programm läuft und alles das tut, was es tun soll – warum um alles in der Welt sollte man es dann optimieren und sich zusätzliche Arbeit machen?

Die zweite Regel ist eine Warnung, mit dem Optimieren zu beginnen, bevor das Programm in seiner ursprünglichen Fassung lauffähig ist.

Wenn Sie alle hier beschriebenen Regeln und Methoden bei der Suche von Fehlern wirklich beherzigen, dann sollte es Ihnen auch als Programmierneuling möglich sein, selbstständig Fehler in Ihren Programmen auszumerzen. (ev)

Relative Dateien leicht verständlich

Die Datenverwaltung zählt – neben dem Spielen – wohl zu den häufigsten Anwendungsgebieten für Heimcomputer. Um Daten aber effektiv verwalten zu können, bedarf es des sinnvollen Einsatzes bestimmter Verfahren und Techniken.

Die relative Dateiorganisation, die sich vor allem bei der Verwaltung größerer Datenbestände als sehr nützlich erweist, ist eine davon und wird im folgenden Artikel näher erläutert.

Zunächst ein paar Worte zum Begriff der »Datei«. Eine Datei läßt sich am besten vergleichen mit einem Karteikasten mit vielen gleichartigen Karteikarten. Einer Karteikarte entspricht dabei ein sogenannter »Datensatz« der Datei. Jeder Datensatz unterteilt sich wiederum in verschiedene »Datenfelder«, in die die einzelnen Eintragungen auf der Karteikarte erfolgen.

Ein kleines Beispiel: In Bild 1 links sehen Sie, wie ein Eintragungs- bzw. Eingabeschema zur Verwaltung von Adressen aussehen könnte. Die Punkte zwischen den Ausrufezeichen markieren die Stellen, an die die einzelnen Daten jeweils geschrieben werden, wie Sie in Bild 1 rechts sehen können. Die Bereiche zwischen den Ausrufezeichen stellen also die einzelnen Datenfelder dar! Alle Inhalte der Datenfelder zusammen (das heißt eine komplette Adresse bestehend aus Nachname, Vorname, Straße, Wohnort und Telefon) ergeben dann einen Datensatz; mehrere gleichartige Datensätze eine Datei. (Eine schematische Darstellung dieses Sachverhalts finden Sie in Bild 2.)

Um nun Daten mit dem Computer zu verwalten, gibt es beim Commodore 64 in Verbindung mit der Floppy VC 1541 grundsätzlich zwei Dateiarten: Die sequentielle und eben die relative.

Bei der sequentiellen Dateiorganisation werden alle Datensätze hintereinander, das heißt sequentiell gespeichert. Wenn nun Datensätze (in unserem Beispiel Adressen) gelesen, hinzugefügt, gelöscht oder geändert werden sollen – diese Vorgänge bezeichnet man übrigens als Dateiarbeit oder »Datenpflege« –, dann muß dazu die komplette Datei in den Rechner geladen und nach Beendigung der Datenpflege wieder auf die Diskette zurückgespeichert werden. Ein direkter Zugriff auf jeden einzelnen Datensatz der Datei ist nicht möglich.

Dieser Umstand ist einerseits sehr zeitraubend (die Floppy 1541 ist ja nicht gerade die Schnellste), andererseits macht er die maximale Dateigröße vom verfügbaren Speicherplatz im Rechner abhängig. Die sequentielle Datenspeicherung ist daher mehr etwas für kleinere Dateien (mit wenigen Datensätzen) oder für Dateien mit sehr kurzen Datensätzen (was wir uns später noch zunutze machen werden).

Wesentlich flexibler in Bezug auf die Datenpflege ist die relative Dateiorganisation, die ja der eigentliche Gegenstand dieses Artikels ist. Bei ihr ist es nämlich möglich, direkt auf jeden einzelnen Datensatz – auch »Record« genannt – der

Nachname: !.....!
 Vorname: !.....!
 Strasse: !.....!
 Wohnort: !.....!
 Telefon: !.....!

Nachname: !Mueller.....!
 Vorname: !Juergen.....!
 Strasse: !Waldstr. 1.....!
 Wohnort: !Frankfurt 1.....!
 Telefon: !1234/677889.....!

Ein Eintrags- beziehungsweise Eingabeschema
 zur Verwaltung von Adressen . . .

. . . und eine Beispieleintragung.

Bild 1. Eine Adressenverwaltung als Beispiel einer Datei

Der Aufbau einer Datei:								
(allgemein)								
Feld1	Feld2	Feld3	Feld4	Feld5	Feld1	Feld2	Feld3	
	Datensatz5				Datensatz6			
DATEI								
(im Beispiel)								
Haus	Hans	Waldstr. 2	Astadt	123456	Mueller	Juergen	Waldstr. 1	
	Adresse5				Adresse6			
ADRESSEN								

Bild 2. So ist eine Datei im Prinzip aufgebaut

Datei zuzugreifen! (Zusätzlich kann sogar auf eine beliebige Stelle innerhalb eines Records positioniert werden.)

Zum Prinzip der relativen Datei

Bei der relativen Dateioorganisation werden alle Datensätze (Records) in laufender Reihenfolge durchnummeriert. Der erste Datensatz bekommt dabei die Nummer 1, der zweite die Nummer 2, etc. Wenn Sie jetzt auf einen Datensatz zugreifen wollen, müssen Sie nur noch dessen Nummer angeben.

Dieser Vorteil ist allerdings mit einer wichtigen Bedingung verbunden:

Alle Records einer Datei müssen eine konstante, beim Einrichten der Datei festzulegende Länge zwischen 1 und 254 Byte (Zeichen) aufweisen!

Diese Bedingung wird verständlich, wenn man sich überlegt, wie das Floppy-Betriebssystem (kurz: Floppy-DOS) aus der Recordnummer die Position des Datensatzes auf der Diskette bestimmt.

Angenommen, Sie wollen auf den 27. Record einer Datei mit einer Recordlänge von 50 Byte zugreifen. Um nun die Position dieses Datensatzes zu ermitteln, multipliziert das Floppy-DOS die Recordlänge (in unserem Fall also 50) mit der Recordnummer -1 (also 26), was die Anzahl der Byte ergibt, die von den ersten 26 Records der Datei belegt werden (=1300 Byte). Dieser Wert wird jetzt noch zur Anfangsposition der Datei hinzuaddiert, was schließlich die Position des gewünschten Datensatzes ergibt. (Die ersten 26 Records werden durch das Aufaddieren ihrer Gesamtlänge auf den Anfang der Datei praktisch »übersprungen«.)

Jetzt wird auch der Name dieser Dateiart, nämlich »relativ«, verständlich, denn die Record-Position wird ja relativ zum

Anfang der Datei ermittelt! (Im Beispiel beginnt der gewünschte Datensatz 1300 Byte vom Anfang der Datei entfernt.)

Wie gesagt kann also jeder Datensatz einer relativen Datei durch Angabe seiner Nummer direkt angesprochen werden. Wenn Sie aber zum Beispiel eine bestimmte Adresse suchen, werden Sie in der Regel kaum die Nummer des betreffenden Datensatzes kennen. Das gleiche gilt auch für die meisten anderen Anwendungsgebiete. (Höchstens bei einer Kontenverwaltung – zum Beispiel für ein Haushaltsbuch – wäre eine direkte Verwendung der Recordnummer sinnvoll.) Daher verwendet man in der Praxis häufig eine Mischform aus sequentieller und relativer Dateioorganisation, die »indexsequentielle Datei«.

Das Prinzip ist ganz einfach:

Alle Daten werden – wie bisher – in einer relativen Hauptdatei abgelegt. Zusätzlich wird dann eine sequentielle Index- oder Schlüsseldatei angelegt. Diese Schlüsseldatei enthält von jedem Datensatz der relativen Datei nur einen Teil – bei einer Adreßdatei zum Beispiel den Nachnamen – sowie die Recordnummer des zugehörigen Datensatzes der relativen Hauptdatei, in dem – im Beispiel – die komplette Adresse untergebracht ist.

Wenn Sie jetzt beispielsweise nach der Adresse eines Herrn Müller suchen, wird einfach die Schlüsseldatei komplett in den Computer geladen (da sie sehr kurz ist, geht dies ja relativ schnell, und auch die Maximalanzahl der Adressen ist verhältnismäßig hoch), dort durchsucht und – sofern der Datensatz »Müller« vorhanden ist – auf den bei »Müller« (in Form der Recordnummer) vermerkten Datensatz positioniert. Dieser wird dann schließlich in den Speicher geladen und ausgegeben.

Natürlich ist es auch möglich, zu einer relativen Hauptdatei mehrere Indexdateien anzulegen (zum Beispiel eine mit den Nachnamen als Index, eine mit den Vornamen, etc.). Allerdings muß man dann das eventuell wieder auftretende Speicherplatzproblem – die Indexdateien müssen ja komplett in den Speicher geladen werden – im Auge behalten beziehungsweise muß sich vor der Dateiarbeit entscheiden, mit welcher Indexdatei man jeweils arbeiten will.

Nach all der Theorie kommen wir nun zur Praxis:

Wie werden relative Dateien mit dem Commodore 64 und der Floppy VC 1541 – die Anlage relativer Dateien auf Kassette ist nicht möglich! – angelegt und verwaltet? Mit dieser Frage wollen wir uns im Folgenden ausführlich befassen. Zum Abschluß wird dann noch eine kleine, ausführlich kommentierte indexsequentielle Adreßverwaltung (Listing 2) vorgestellt, die Sie selbst nach Belieben abändern können, um so Ihre neuen Kenntnisse in der Praxis zu erproben.

Die Verwaltung einer relativen Datei gliedert sich grob in vier Abschnitte:

– »Öffnen« der Datei

- Positionieren auf den gewünschten Record
- Lesen/Schreiben des Records
- »Schließen« der Datei.

Da das Basic 2.0 des Commodore 64 leider keine speziellen Befehle zur Unterstützung von relativen Dateien zur Verfügung stellt, bleibt nur der Weg über »OPEN«- und »CHR\$(...)-Sequenzen. Daher sieht das Ganze auf den ersten (und wohl auch zweiten) Blick etwas kompliziert aus. Wenn man sich die erforderlichen Programmteile aber – wie in Listing 1 geschehen – als Unterprogramme schreibt, ist es halb so schlimm.

Kommen wir zum ersten Schritt, dem »Öffnen« der Datei.

Öffnen einer relativen Datei

Bei einer relativen Datei ist es – im Gegensatz zum Beispiel zur sequentiellen Datei – praktisch egal, ob die Datei auf der Diskette schon eingerichtet ist oder nicht oder ob Sie Daten lesen oder schreiben wollen. Der Öffnungsbefehl ist immer derselbe:

```
OPEN LF,GA,SA,DN$+",L,"+CHR$(RL)
```

Die einzelnen Variablen – auch »Parameter« genannt – haben folgende Bedeutung:

LF: Das ist die logische Filenummer der Datei. Sie ist erforderlich, damit der Computer bei späteren Lese- oder Schreibbefehlen (bei denen diese Nummer angegeben werden muß) weiß, aus welcher Datei gelesen oder in welche Datei geschrieben werden soll. LF kann Werte zwischen 1 und 127 annehmen.

GA: Das ist die Geräteadresse der Floppy. Im Normalfall also 8.

SA: Die »Sekundäradresse« oder »Kanalnummer« SA hat auf Floppy-Ebene eine ähnliche Bedeutung wie die logische Filenummer LF auf Computer-Ebene.

Für uns ist aber nur wichtig, daß SA Werte zwischen 2 und 14 annehmen kann.

Wenn Sie mehrere Dateien gleichzeitig geöffnet haben, müssen Sie außerdem darauf achten, daß Sie jeder Datei eine andere Sekundäradresse zuweisen. Wird nämlich eine zweite Datei mit derselben Sekundäradresse wie die erste geöffnet, so wird die zuerst geöffnete Datei auf Floppy-Ebene geschlossen!

In diesem Zusammenhang noch ein wichtiger Hinweis:

Auf Floppy-Ebene können maximal drei Dateien gleichzeitig geöffnet sein. (Auf Computer-Ebene sind es übrigens maximal 10). Da zur Verwaltung einer relativen Datei neben der eigentlichen Datei – wie wir gleich noch sehen werden – auch der Floppy-Kommandokanal geöffnet werden muß, bedeutet das, daß maximal eine relative und eine sequentielle Datei gleichzeitig geöffnet sein können (was gerade für eine indexsequentielle Dateiorganisation reicht).

DN\$: DN\$ enthält den Namen der Datei. Dieser darf maximal 16 Zeichen lang sein. Außerdem ist darauf zu achten, daß kein File (= Datei oder Programm) desselben Namens bereits auf der Diskette existiert – außer natürlich eine früher definierte relative Datei dieses Namens, mit der Sie jetzt arbeiten wollen, denn sonst gibt es einen »FILE EXISTS ERROR«.

„L,“: Dieser Parameter teilt dem Floppy-DOS mit, daß eine relative Datei geöffnet werden soll. (Die Kommas dienen zur Abtrennung von den übrigen Parametern.)

RL: RL enthält die gewünschte Recordlänge der Datei und kann Werte zwischen 1 und 254 annehmen.

Wichtig: Die beim ersten Öffnen der Datei angegebene Recordlänge kann später nicht mehr geändert werden! (Wenn Sie versuchen, eine bereits existierende relative Datei mit einer anderen Recordlänge als bei der ersten Öffnung angegeben zu öffnen, so wird ein »RECORD NOT PRESENT ERROR« angezeigt.)

Zur Ermittlung der Recordlänge zählen Sie einfach alle Datenfeldlängen eines Datensatzes zusammen. Im Eingangsbeispiel ergibt das eine Länge von 15 (Nachname) + 10 (Vorname) + 15 (Straße) + 15 (Wohnort) + 11 (Telefon) = 66 Byte.

Sofern Sie zum Einlesen der Records in den Computer den INPUT #-Befehl benutzen, müssen Sie dazu noch ein Byte addieren, denn, um das Ende eines Records erkennen zu können, benötigt der INPUT #-Befehl ein »RETURN« (entspricht CHR\$(13)) oder ein Komma (entspricht CHR\$(44)) am Ende des Records. Das »RETURN« wird normalerweise am Ende eines jeden PRINT- beziehungsweise PRINT #-Befehls gesendet – sofern Sie es nicht durch ein Semikolon am Ende des PRINT-Kommandos (zum Beispiel bei »PRINT #1, "text";«) unterdrücken.

Im Beispiel ergibt sich also eine Gesamtlänge von 67 Byte. Nachdem die Datei geöffnet wurde, kann mit der eigentlichen Dateiarbeit begonnen werden. Um jetzt einen bestimmten Record zu schreiben beziehungsweise zu lesen, muß zunächst auf ihn positioniert werden.

Positionieren auf einen Record

Da die Positionieranweisung über den Floppy-Kommandokanal gesendet werden muß, muß dieser zuerst mit »OPEN 15,8,15,« geöffnet werden. Die erste 15 ist wieder die logische Filenummer (Um Kollisionen mit Dateien zu vermeiden, ist es zweckmäßig, hier immer dieselbe logische Filenummer zu verwenden); bei der 8 handelt es sich um die Geräteadresse der Floppy. Die zweite 15 ist die Kanalnummer des Floppy-Kommandokanals.

Der Positionierbefehl ist ähnlich kompliziert aufgebaut wie der OPEN-Befehl:

```
PRINT #15,"P"+CHR$(SA)+CHR$(LB)+CHR$(HB)+CHR$(RP)
```

Die Parameter im einzelnen:

SA: Das ist die beim Öffnen der Datei festgelegte Sekundäradresse.

LB/HB: Das ist die Recordnummer in einer für Sie vielleicht etwas ungewohnten Form, der sogenannten Lowbyte-/Highbyte-Form.

Da mit CHR\$(...) nur Werte bis zu 255 übermittelt werden können, muß die Recordnummer in zwei solcher Werte aufgeteilt werden, die sich aus folgenden Formeln ergeben:

$HB = \text{INT}(RN/256); LB = RN - 256 * HB$

RP: Mit RP können Sie auf eine bestimmte Stelle innerhalb des Records positionieren. Wollen Sie zum Beispiel einen Record ab dem 10. Zeichen einlesen, so muß RP den Wert 10 erhalten.

Wichtig: Eine Positionierung auf eine Stelle innerhalb eines Records ist nur beim Lesen dieses Records sinnvoll. Zum Schreiben eines Records muß RP auf 1 gesetzt werden!

Und natürlich ist auch nur eine solche Positionsangabe sinnvoll, die kleiner gleich der Recordlänge ist. Daher ergibt sich für RP – eine entsprechende Recordlänge vorausgesetzt – ein Maximalwert von 254.

Nach der Positionierung ist der betreffende Record nun für einen Lese- oder Schreibzugriff bereit.

Schreiben eines Records

Vor dem Schreiben eines Records sind drei Dinge zu beachten:

- Es muß unbedingt auf das erste Byte des Records positioniert werden (RP=1).
- Es sollte sichergestellt werden, daß alle Datenfelder – und

damit auch der Record – ihre vorgeschriebene Länge aufweisen. Wie eine solche Kontrolle und Korrektur aussehen könnte, sehen Sie in Listing 2 in den Zeilen 860–920.

- Da der Record in einem Stück geschrieben werden muß, müssen die einzelnen Datenfelder vor dem Schreiben zusammengefaßt werden (siehe auch Listing 2 Zeile 940 bis 960).

Sind diese Voraussetzungen erfüllt, so kann der Record mit folgendem Befehl geschrieben werden:

PRINT # LF, RC\$

LF ist dabei die beim OPEN-Befehl festgelegte logische Filenummer; RC\$ enthält den zu schreibenden Record.

In diesem Zusammenhang erwähnenswert ist die Fehlermeldung »RECORD NOT PRESENT ERROR«, was soviel bedeutet wie »Datensatz nicht vorhanden«. Falls nämlich auf einen Record zugegriffen (geschrieben/gelesen) wird, der zuvor noch nicht beschrieben wurde, so wird diese Meldung erzeugt. Beim (erstmaligen) Schreiben eines Records kann diese Meldung natürlich ignoriert werden. (Der Schreibbefehl wird trotz der Meldung ausgeführt.)

Weiterhin ist zu beachten: Ein Schreibbefehl auf einen zuvor noch nicht beschriebenen Record hat zur Folge, daß alle davor liegenden (zuvor noch nicht beschriebenen) Records ebenfalls beschrieben werden!

Ein Beispiel: Sie haben eine relative Datei bereits bis zum 20. Record beschrieben. Nun schreiben Sie den 100. Record. Dadurch werden gleichzeitig die Records 21 bis 99 (mit demselben Inhalt wie der 100. Record) beschrieben!

Wenn Sie also eine relative Datei nicht der Reihe nach beschreiben wollen, dann sollten Sie beim Einrichten der Datei als erstes den Record mit der höchsten Nummer (mit einem beliebigen Inhalt) beschreiben. Dadurch bleiben Ihnen unnötige Wartezeiten bei der Dateiarbeit – das Beschreiben von zum Beispiel 50 Records nimmt einige Zeit in Anspruch – erspart.

Lesen eines Records

Zum Lesen eines Records gibt es zwei Möglichkeiten:

Die erste Möglichkeit stellt der INPUT #-Befehl dar. Mit ihm kann ein Record komplett in den Computer geladen werden – unter zwei Bedingungen: Der Record darf nicht länger als 88 Zeichen sein, und er muß ein »RETURN« oder ein Komma als letztes Zeichen aufweisen.

Das Befehlsformat: INPUT # LF, RC\$

LF ist dabei die beim OPEN-Befehl festgelegte logische Filenummer; RC\$ wird der Record zugewiesen.

Eine zweite Möglichkeit – bei einer Recordlänge von mehr als 88 Byte zudem die einzige – bietet der GET #-Befehl. Um zum Beispiel einen 100 Byte langen Record einzulesen, wäre folgende Befehlssequenz denkbar:

FOR I=1 TO 100:GET # LF,EG\$:RC\$=RC\$+EG\$:NEXT I

LF ist dabei wieder die beim OPEN-Befehl festgelegte logische Filenummer.

Schließen einer relativen Datei

Nach Beendigung der Dateiarbeit muß die Datei geschlossen werden. Der dazu erforderliche Befehl ist – ausnahmsweise – sehr einfach.

CLOSE LF

LF ist dabei wiederum die beim OPEN-Befehl festgelegte logische Filenummer.

Zum Abschluß eine kleine Adreßverwaltung zum Experimentieren. Bitte geben Sie dazu Listing 2 ein.

Wichtig: Das Programm ist nur mit den in Listing 1 enthaltenen Unter-Programmen lauffähig. Tippen Sie also am besten zuerst Listing 1 ein und speichern es (für spätere eigene Entwicklungen, für die Sie die Unterprogramme gut gebrauchen können). Danach geben Sie zusätzlich Listing 2 ein und speichern das Gesamtprogramm dann ebenfalls auf Diskette.

Da das Programm selbst ausführlich dokumentiert ist, möchte ich an dieser Stelle nur noch kurz ein paar Bedienungshinweise geben:

Nachdem das Hauptmenü ausgegeben wurde, geben Sie als erstes »1« (+ »RETURN«) ein. Dadurch werden alle notwendigen Vorbereitungen für die Dateiarbeit getroffen. Sobald das Hauptmenü wieder erschienen ist, können Sie wahlweise (und auch abwechselnd) Adressen ein- oder ausgeben. Bei der Ausgabe wird dabei der Nachname als Index verwendet. Nach Beendigung der Dateiarbeit geben Sie »4« ein, wodurch die Datei ordnungsgemäß geschlossen und das Programm beendet wird.

Das Programm ist zwar recht spartanisch, dafür aber sehr flexibel. Beides soll Sie dazu animieren, selbst tätig zu werden und das Programm nach Ihren speziellen Wünschen abzuändern und zu ergänzen. Denn auch bei der Datenverwaltung gilt, wie überall: Übung macht den Meister!

(Martin Hecht/gk)

```

1680 REM--- UNTERPROGRAMME ZUR VERWALTUNG
      VON RELATIVEN DATEIEN -----
-
1740 :                                     <016>
1750 REM  LF = LOGISCHE FILENUMMER      <192>
1760 REM  SA = SEKUNDAERADRESSE         <152>
1770 REM  DN$= DATEINAME                <169>
1780 REM  RL = RECORDLAENGE             <120>
1790 :                                     <242>
1800 OPEN 15,8,15:REM FLOPPY-KOMMANDOKANAL
      OEFFNEN                           <242>
1810 OPEN LF,8,SA,DN$+"",L,"+CHR$(RL):REM D
      ATEI OEFFNEN                       <163>
1820 :                                     <041>
1830 RETURN                             <018>
1840 :                                     <110>
1850 :                                     <038>
1860 :                                     <048>
1870 :                                     <058>
1870 REM--- POSITIONIEREN AUF EINEN RECORD
      -----
-
1880 :                                     <020>
1890 REM  SA = SEKUNDAERADRESSE         <078>
1900 REM  RN = RECORDNUMMER             <045>
1910 REM  RP = POSITION INNERHALB DES RECO
      RDS                                <197>
1920 :                                     <167>
1930 HB=INT(RN/256):LB=RN-256*HB:REM RECO
      DNUMBER IN LOW/HIGH AUFTEILEN     <118>
1940 PRINT#15,"P"+CHR$(SA)+CHR$(LB)+CHR$(H
      B)+CHR$(RP)                       <060>
1950 :                                     <089>
1960 RETURN                             <148>
1970 :                                     <240>
1980 :                                     <168>
1990 :                                     <178>
2000 REM--- SCHLIESSEN EINER (RELATIVEN) D
      ATEI -----
-
2010 :                                     <088>
2020 REM  LF = LOGISCHE FILENUMMER      <208>
2030 :                                     <168>
2040 CLOSE LF:REM RELATIVE DATEI SCHLIESSE
      N                                 <228>
2050 CLOSE 15:REM FLOPPY-FEHLERKANAL SCHLI
      ESSEN                             <234>
2060 :                                     <171>
2070 RETURN                             <004>
                                     <096>

```

© 64'er

Listing 1. Unterprogramme zur Verwaltung einer relativen Datei

```

100 REM--- ADRESSVERWALTUNG MIT INDEXSEQUENTIELLER D
ATEIORGANISATION ----- <217>
110 REM--- (W) 1985 BY MARTIN HECHT,STGT ----- <132>
----- <096>
120 : <106>
130 : <116>
140 :
150 REM--- DIMENSIONIERUNGEN ----- <043>
----- <136>
160 : <056>
170 DM=100:REM MAXIMALE DATENSATZANZAHL <156>
180 : <146>
190 DIM ID$(DM):REM INDEXFELD (ENTHAELT NACHNAMEN)
200 DIM IN(DM):REM ENTHAELT ZUGEHÖRIGE RECORDNUMMER
N <086>
210 DIM DS$(5):REM DATENFELDER <063>
220 : <196>
230 : <206>
240 : <216>
250 REM--- HAUPTMENUE ----- <249>
----- <238>
260 : <199>
270 WL=0:PRINT:PRINT <039>
280 PRINT TAB(11) "ADRESSENVERWALTUNG" <138>
290 PRINT <048>
300 PRINT TAB(8)"-1-: DATEIARBEIT BEGINNEN" <173>
310 PRINT TAB(8)"-2-: ADRESSEN EINGEBEN" <250>
320 PRINT TAB(8)"-3-: ADRESSEN AUSGEBEN" <108>
330 PRINT TAB(8)"-4-: DATEIARBEIT BEENDEN" <188>
340 PRINT <072>
350 : <166>
360 PRINT TAB(11) "IHRE WAHL (1-4)";:INPUT WL <092>
370 : <180>
380 IF WL=1 THEN GOSUB 490:REM DATEIARBEIT BEGINNEN <036>
390 IF WL=2 THEN GOSUB 710:REM ADRESSEN EINGEBEN <132>
400 IF WL=3 THEN GOSUB 1110:REM ADRESSEN AUSGEBEN <099>
410 IF WL=4 THEN GOTO 1480:REM DATEIARBEIT BEENDEN <142>
420 : <123>
430 GOTO 270:REM ZURUECK ZUM HAUPTMENUE <162>
440 : <172>
450 : <182>
460 :
470 REM--- DATEIARBEIT BEGINNEN ----- <005>
----- <202>
480 :
490 LF=1:SA=2:DN$="RELADR":RL=67:GOSUB 1800:REM RELA
TIVE DATEI 'RELADR' OEFFNEN <134>
500 : <222>
510 OPEN 2,8,3,"SEQADR,S,R":REM SEQUENTIELLE DATEI
SEQADR' ZUM LESEN OEFFNEN <156>
520 : <244>
530 GOSUB 1620:REM FLOPPY-FEHLERKANAL AUSLESEN <096>
540 IF ER<>0 THEN AD=0:GOTO 590:REM DATEI IST NOCH N
ICHT ANGELEGT <154>
550 : <018>
560 INPUT#2,AD:REM ANZAHL DER DATENSATZETZ <204>
570 FOR I=1 TO AD:INPUT#2,ID$(I),IN(I):NEXT I:REM IN
DEXDATEI EINLESEN <101>
580 : <048>
590 FL=1:REM KENNZEICHNUNG FUER 'DATEI IM RECHNER' <189>
600 : <068>
610 CLOSE 2:REM SEQUENTIELLE DATEI SCHLIESSEN <147>
620 : <088>
630 PRINT:PRINT:PRINT TAB(7) "DATEIARBEIT KANN BEGIN
NEN !" <032>
640 : <108>
650 RETURN <200>
660 : <128>
670 : <138>
680 : <148>
690 REM--- ADRESSEN EINGEBEN ----- <049>
----- <168>
700 :
710 IF FL=0 THEN PRINT:PRINT TAB(8) "KEINE DATEI IM
RECHNER !":RETURN <110>
720 : <188>
730 REM ADRESSE EINGEBEN <009>
740 PRINT:PRINT <184>
750 INPUT "NACHNAME (15) :";DS$(1) <225>
760 INPUT "VORNAME (10) :";DS$(2) <135>
770 INPUT "STRASSE (15) :";DS$(3) <123>
780 INPUT "WOHNORT (15) :";DS$(4) <160>
790 INPUT "TELEFON (11) :";DS$(5) <078>
800 : <119>
810 REM ADRESSE IN INDEXDATEI VERMERKEN <065>
820 AD=AD+1:REM ANZAHL DER ADRESSEN UM 1 ERHOEHEN <107>
830 ID$(AD)=DS$(1):REM NACHNAME <135>
840 IN(AD)=RN:REM RECORDNUMMER <131>
850 : <064>
860 REM ADRESSFELDER AUF RICHTIGE LAENGE BRINGEN <218>
870 LE$="(15SPACE)":REM LEERSTRING ZUM AUFFUELLEN DE
R FELDER <190>
880 DS$(1)=LEFT$(DS$(1)+LEFT$(LE$,ABS(15-LEN(DS$(1)
)),15) <234>
890 DS$(2)=LEFT$(DS$(2)+LEFT$(LE$,ABS(10-LEN(DS$(2)
)),10) <129>
900 DS$(3)=LEFT$(DS$(3)+LEFT$(LE$,ABS(15-LEN(DS$(3)
)),15) <143>
910 DS$(4)=LEFT$(DS$(4)+LEFT$(LE$,ABS(11-LEN(DS$(4)
)),11) <099>
920 DS$(5)=LEFT$(DS$(5)+LEFT$(LE$,ABS(11-LEN(DS$(5)
)),11) <005>
930 : <144>
940 REM ADRESSE FUER SPEICHERUNG ZUSAMMENFASSEN <129>
950 RC$="" <080>
960 FOR I=1 TO 5:RC$=RC$+DS$(I):NEXT I <083>
970 : <184>
980 REM ADRESSE SPEICHERN <233>
990 RN=AD:RP=1:GOSUB 1930:REM AUF RECORD (NR.=AD) PO
SITIONIEREN <142>
1000 PRINT#LF,RC$:REM ADRESSE IN RECORD SCHREIBEN <249>
1010 GOSUB 1600:REM FLOPPY-FEHLERKANAL AUSLESEN <002>
1020 : <234>
1030 PRINT:PRINT:PRINT TAB(5) "ADRESSE IST GESPEICHER
T !" <040>
1040 : <080>
1050 RETURN <092>
1060 : <020>
1070 : <030>
1080 : <040>
1090 REM--- ADRESSEN AUSGEBEN ----- <207>
----- <060>
1100 :
1110 IF FL=0 THEN PRINT:PRINT TAB(8) "KEINE DATEI IM
RECHNER !":RETURN <002>
1120 : <080>
1130 PRINT:PRINT:INPUT "NACHNAME";NN$:REM INDEX ERFR
AGEN <201>
1140 : <100>
1150 FOR I=1 TO AD:REM INDEXDATEI DURCHSUCHEN <115>
1160 IF NN$=ID$(I) THEN GN=I:I=AD:NEXT I:GOTO 1220:R
EM INDEX GEFUNDEN <196>
1170 NEXT I <238>
1180 : <140>
1190 PRINT:PRINT:PRINT TAB(5) "ADRESSE IST NICHT VOR
HANDEN !" <212>
1200 RETURN <242>
1210 : <170>
1220 RN=GN:RP=1:GOSUB 1930:REM AUF RECORD (NR. = IN(
I)) POSITIONIEREN <184>
1230 INPUT#LF,RC$:REM RECORD EINLESEN <029>
1240 : <200>
1250 REM RECORD AUFTHEILEN <005>
1260 DS$(1)=MID$(RC$,1,15):REM NACHNAME <158>
1270 DS$(2)=MID$(RC$,16,10):REM VORNAME <080>
1280 DS$(3)=MID$(RC$,26,15):REM STRASSE <133>
1290 DS$(4)=MID$(RC$,41,15):REM WOHNORT <155>
1300 DS$(5)=MID$(RC$,56,11):REM WOHNORT <074>
1310 : <016>
1320 REM RECORD AUSGEBEN <011>
1330 PRINT:PRINT <012>
1340 PRINT "NACHNAME: ";DS$(1) <025>
1350 PRINT "VORNAME: ";DS$(2) <089>
1360 PRINT "STRASSE: ";DS$(3) <111>
1370 PRINT "WOHNORT: ";DS$(4) <087>
1380 PRINT "TELEFON: ";DS$(5) <073>
1390 : <096>
1400 PRINT:PRINT:PRINT "WEITER MIT 'SPACE' !" <131>
1410 GET T$:IF T$<>CHR$(32) THEN 1410:REM WARTEN AUF
'RETURN' <069>
1420 : <126>
1430 RETURN <218>
1440 : <146>
1450 : <156>
1460 : <166>
1470 REM--- DATEIARBEIT BEENDEN ----- <022>
-----
1480 PRINT#15,"S:SEQADR":REM INDEXDATEI AUF DISKETTE
LOESCHEN <222>
1490 : <196>
1500 OPEN 2,8,3,"SEQADR,S,W":REM SEQ. DATEI 'SEQADR'
ZUM SCHREIBEN OEFFNEN <111>
1510 PRINT#2,AD:REM ANZAHL DER DATENSATZETZ <216>
1520 FOR I=1 TO AD:PRINT#2,ID$(I),"IN(I):NEXT I:REM
INDEXDATEI SPEICHERN <192>
1530 : <236>
1540 GOSUB 2040:REM RELATIVE DATEI SCHLIESSEN <250>
1550 : <002>
1560 END:REM PROGRAMM BEENDEN <250>
1570 : <022>
1580 : <032>
1590 : <042>
1600 REM--- FLOPPY-FEHLERKANAL AUSLESEN ----- <223>
-----
1610 : <062>
1620 INPUT#15,ER,ER$,TR,SK <076>
1630 : <082>
1640 RETURN <174>
1650 : <102>
1660 : <112>
1670 : <122>

```

0 64'er

Listing 2. Eine einfache Adreßverwaltung als Beispiel für eine indexsequentielle Datei